Visual Programming

Date: 08 / December / 2025

Submitted To: Engr. Raheela Ambrin

Submitted From		
Name	**Enrollment**	**Class**
Mudassir Riaz	01-131232-046	BSE 5B
M. Sami Shahid	01-131232-066	BSE 5B

SYSTEM REQUIREMENTS SPECIFICATION

1. Introduction

The purpose of this document is to define the Software Requirements Specification (SRS) for the **FYP Automation System**. This system is designed to digitize the entire lifecycle of Final Year Projects at Bahria University, replacing manual paper-based workflows with a centralized web portal. It covers registration, proposal defense, monitoring, financial budgeting, and final grading.

1.1. Product Scope

The **FYP Automation System** is a web-based platform that connects Students, Supervisors, the FYPCoordinator, and the Examination Cell. It streamlines the management of academic projects frominception to archiving.

1.2. Objectives & Goal

- **Eliminate Paperwork:** Remove the need for physical Forms A, B, C, and D.
- **Real-time Tracking:** Provide a live dashboard for the coordinator (e.g., Mam Sadaf) to monitor group statuses.
- **Centralized Grading:** Automate the calculation of marks from proposal defenses, monthly logs, and final vivas.
- **Financial Transparency:** automate the request and disbursement of project funding.

2. Overall Description

The FYP Automation System is a new, standalone product. It will replace the current manual system of Excel sheets and physical files. It is designed to integrate with the university's existing email infrastructure (SMTP) and potentially the Learning Management System (LMS) for authentication in the future.

2.1. Product Functions

The system provides the following core functions:

- Student Registration & Group Formation.
- Proposal Submission & Supervisor Allocation.
- Monthly Progress Logging & Grading.
- Defense Scheduling & Digital Rubrics.
- Financial Budget Request & Audit.
- Final Thesis Archiving.

2.2. User Classes and Characteristics

Role	Login Method	Main Capabilities
SuperAdmin	Username/Password (Users table)	All Admin functions + HOD access, System configuration
HOD	Username/Password (Staff.Username → User)	View escalations, Approve budgets, Review compiled marks, Oversee department FYPs
FYP Coordinator	Username/Password (Staff.Username → User)	Manage groups, Review proposals (Form-A/B/C), Unlock documents, Schedule defenses, Compile results
Supervisor/Teacher	Username/Password (Staff.Username → User)	Accept/reject supervision requests, Review log forms, Submit Form-D, Enter student marks
Finance Officer	Username/Password (Staff.Username → User)	Budget disbursements (planned feature)
Student	Enrollment ID/Password (Students table)	Create/join groups, Submit forms (A/B/C), Upload documents, View defenses, Track progress

2.3. Operating Environment

- **Client:** Modern Web Browsers (Chrome, Edge, Safari).
- **Server:** Windows Server / Linux.
- **Database:** Microsoft SQL Server.
- **Backend Framework:** ASP.NET Core 8 Web API.
- **Frontend Framework:** React.js.

2.4. Design & Implementation Constraints

- **Enrollment ID Format:** Must strictly follow XX-XXXXXX-XXX.
- **Internet Access:** Users are assumed to have continuous internet connection.

2.5. Assumptions & Dependencies

- It is assumed that the University Finance Department will accept digital approval from the HOD.
- It is assumed that all students have valid university email addresses.

3. System Features

3.1 User Management

3.1.1 User Registration: The System Admin can bulk import users or register them manually. The system validates theEnrollment ID format for students.

3.1.2 User Login: Role-based login using Username/Enrollment ID and Password. The system redirects users to theirspecific dashboard upon login.

3.2 Proposal Submission & Allocation

3.2.1 Submission Slots: The Coordinator creates a time-bound slot for proposals.

3.2.2 Form-A & Form-B: Students submit group details and select a supervisor. The system updates the dashboard status to "Waiting for Approval".

3.2.3 Digital Endorsement (Form-D): Supervisors can digitally sign the confirmation letter, which officially registers the group in the system.

3.3 Monitoring & Evaluation

3.3.1 Monthly Logs: Students submit weekly logs; Supervisors assign a grade (0-10). Missed logs trigger an auto-escalationto the HOD.

3.3.2 Defense Scheduling: The Coordinator schedules defenses. The system notifies all parties of the date and venue.

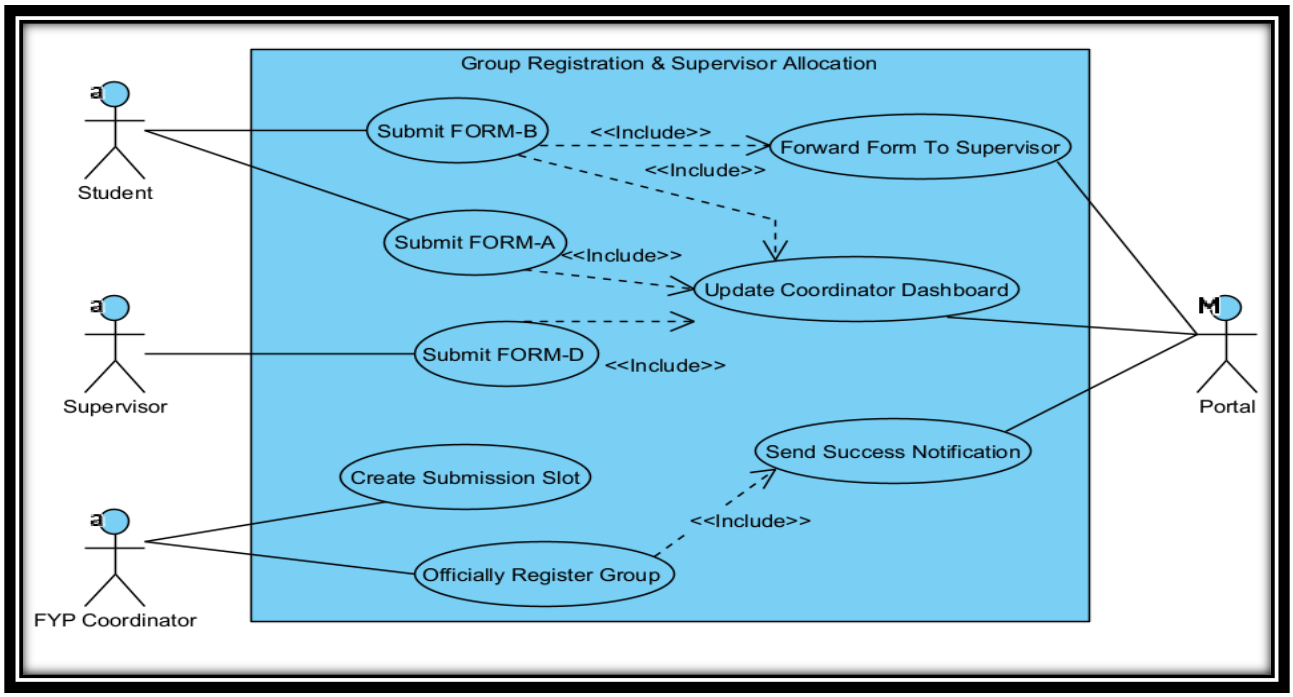
3.4 Finance & Archiving

4.4.1 Budget Requests: Students upload Bill of Quantities (BoQ). The HOD approves the amount, and the Finance Officermarks it as "Disbursed".

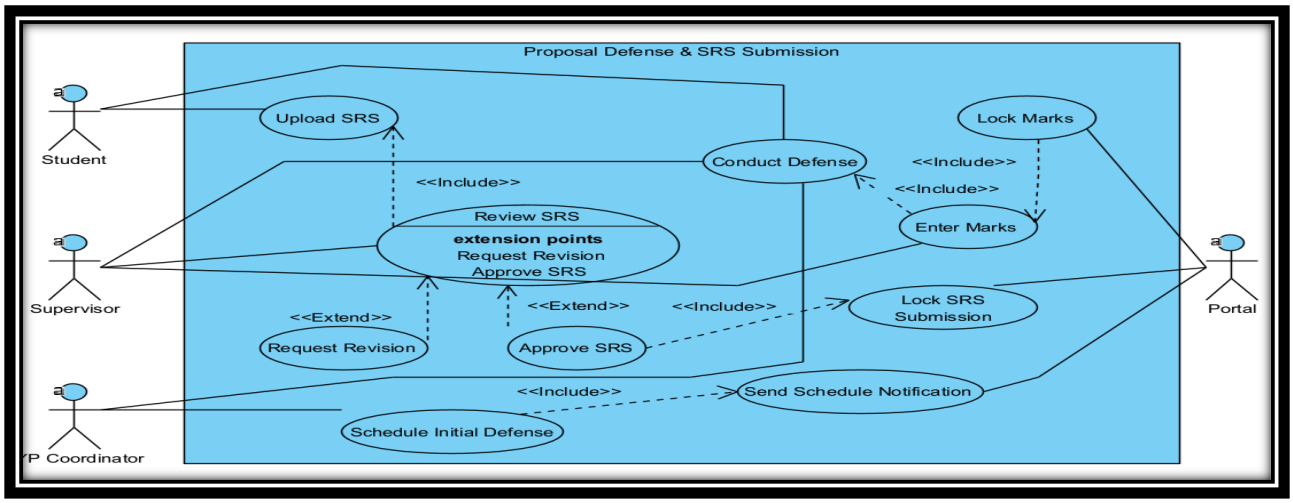
4.4.2 Project Archiving: The Exam Cell locks the project state after the final viva and generates the closure report.

4. Use Cases Diagram

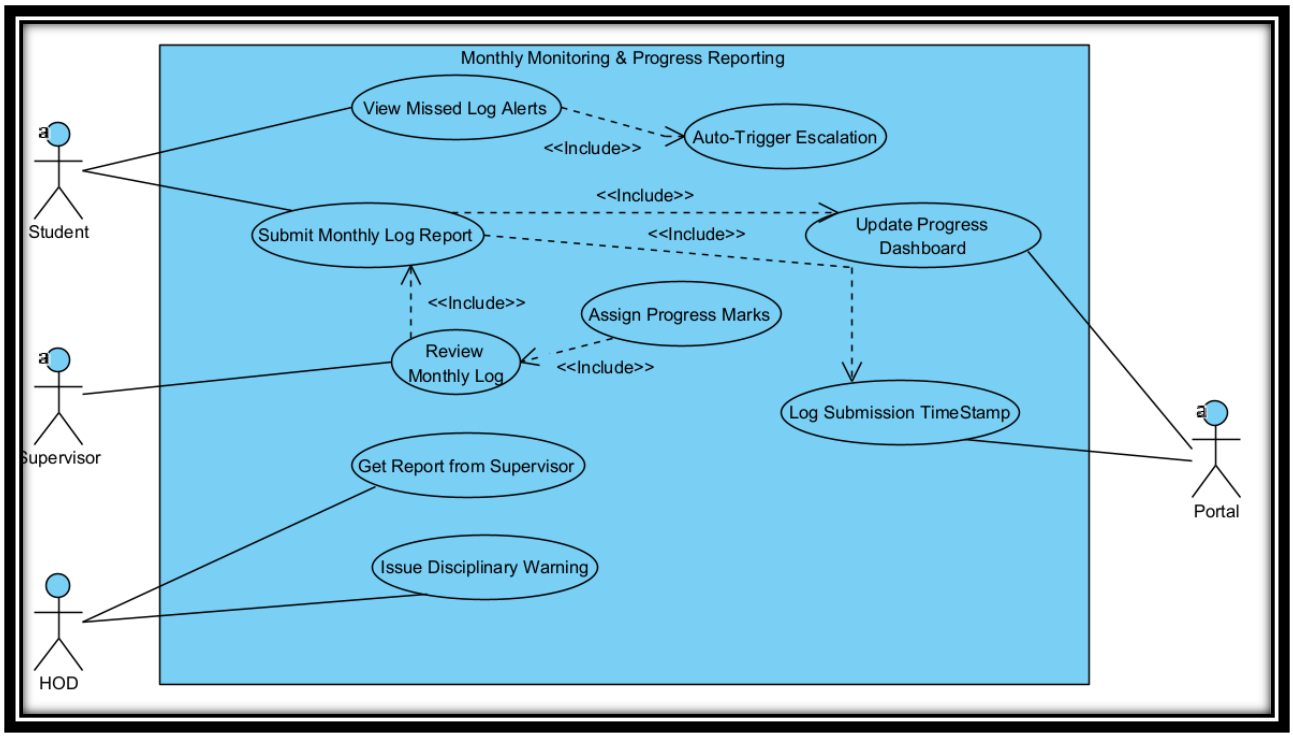
- **Group Registration & Allocation:**



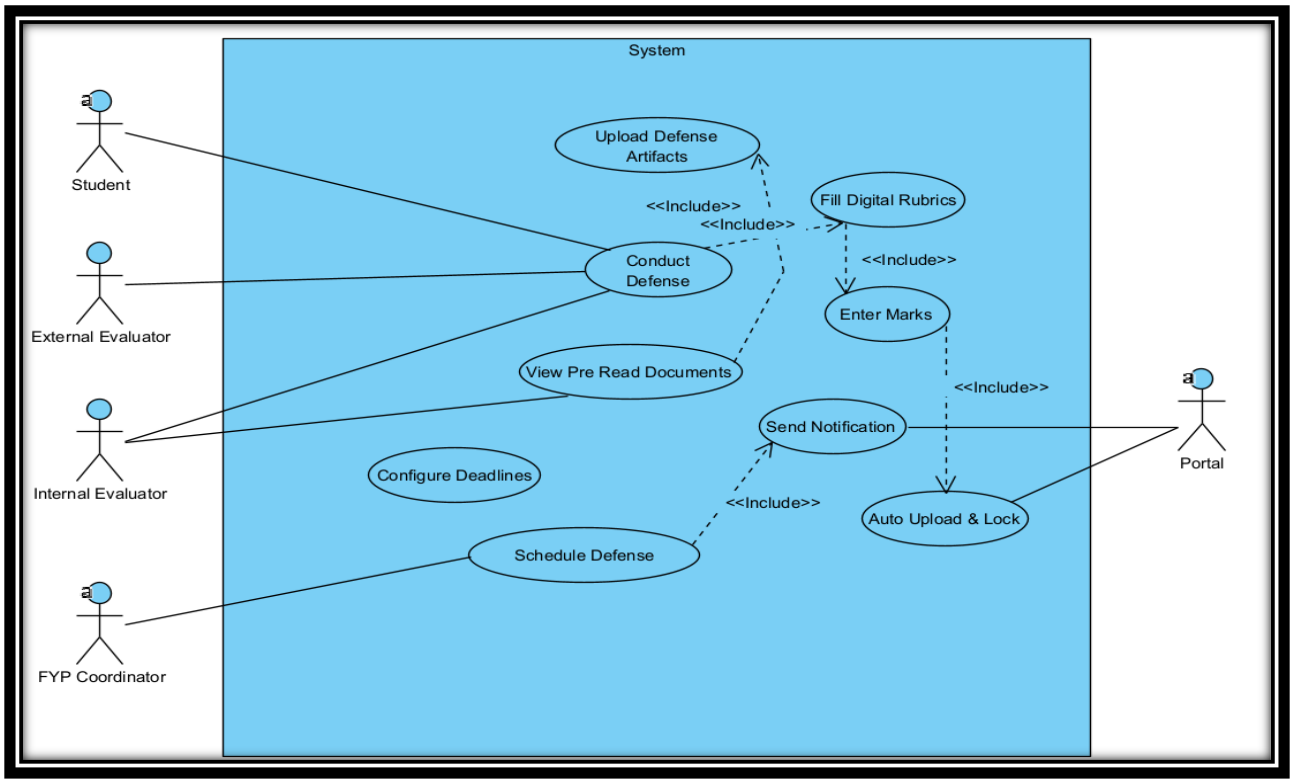
- **Proposal Submission & SRS Submission:**



- **Monthly Monitoring & Progress Reporting:**

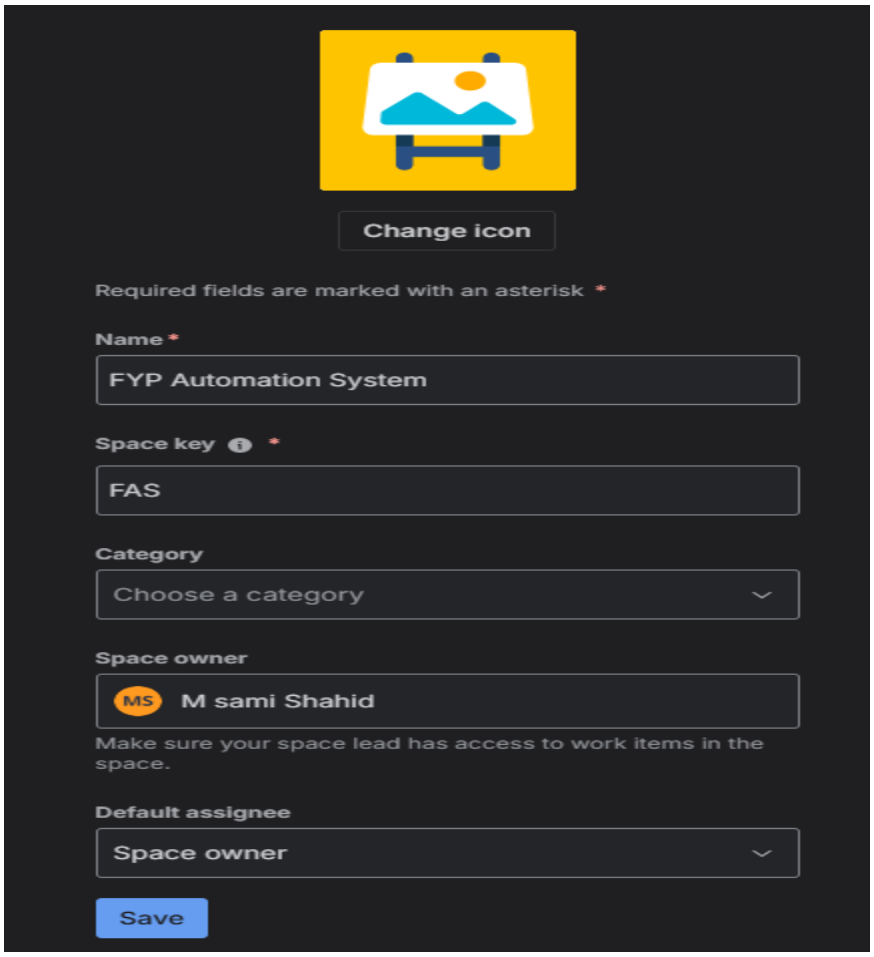


- **Defense & Evaluation:**

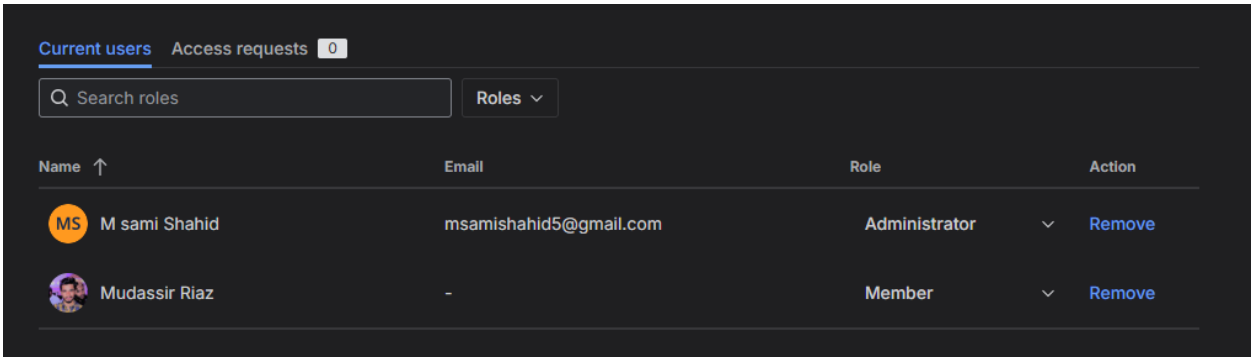


5. User Stories

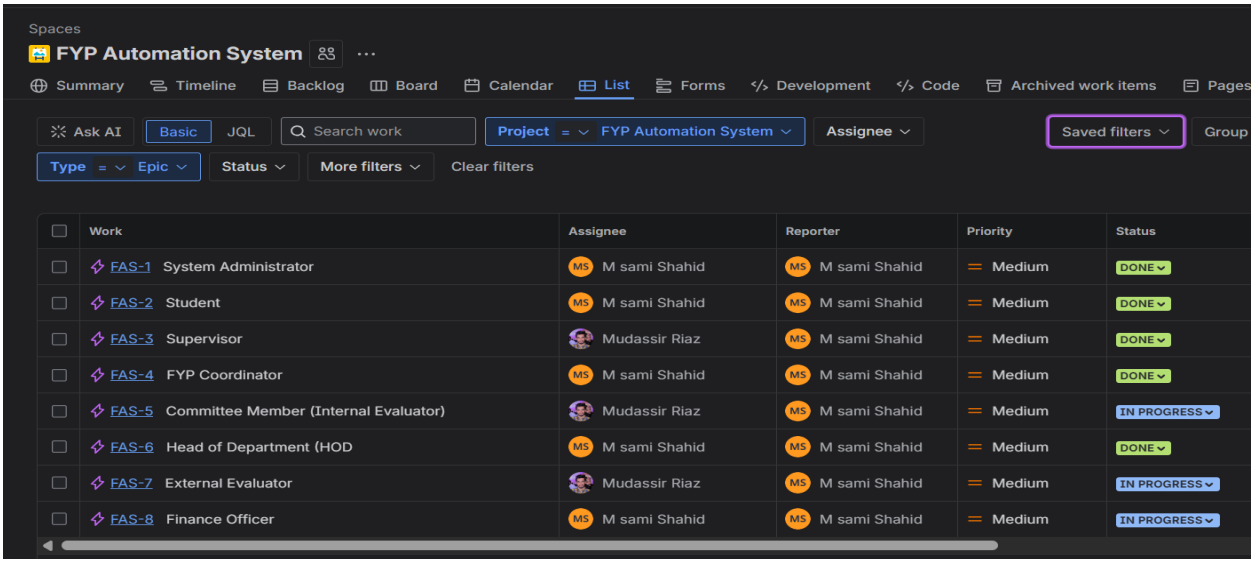
- **Jira Project Space:**



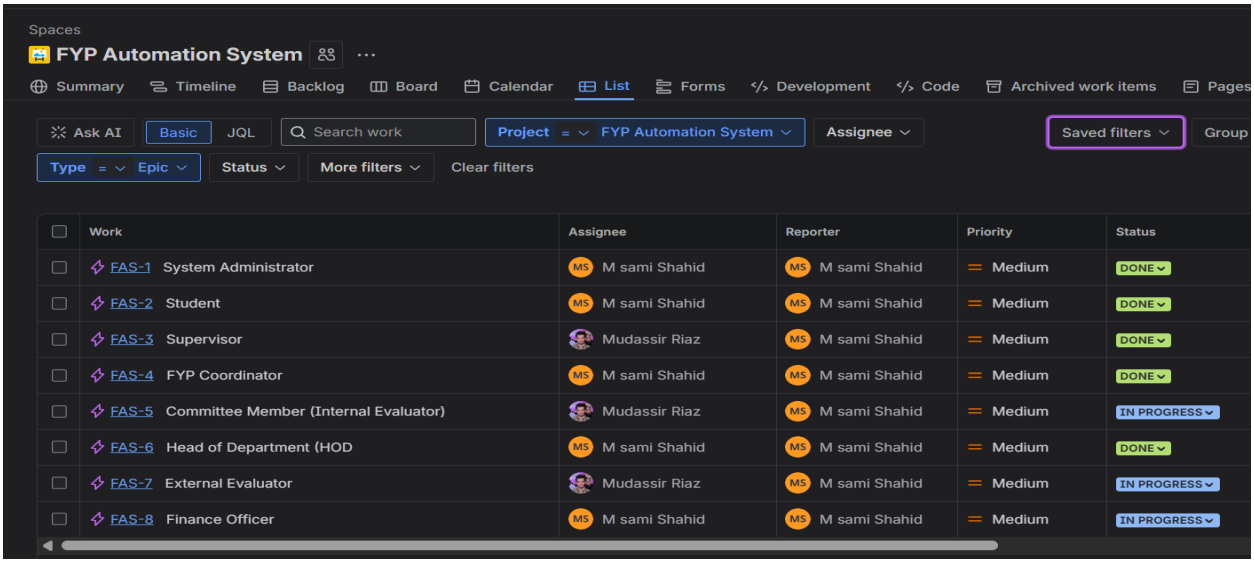
- **Access:**



- **List of Epics:**



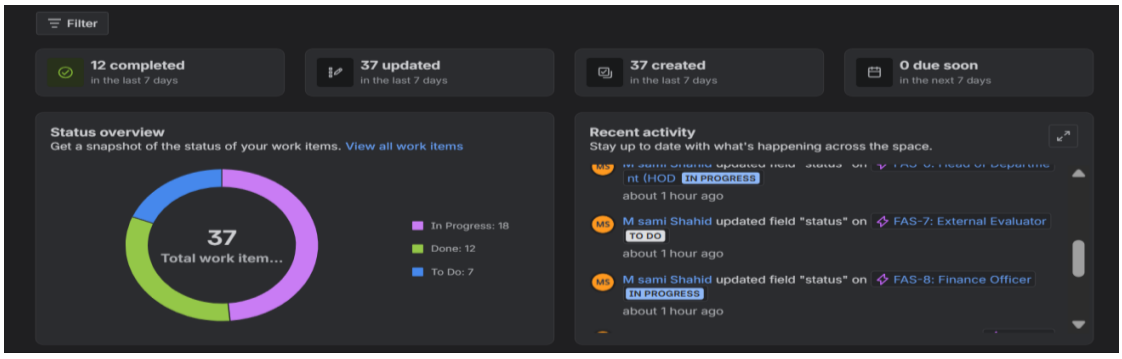
- **List of Epics:**

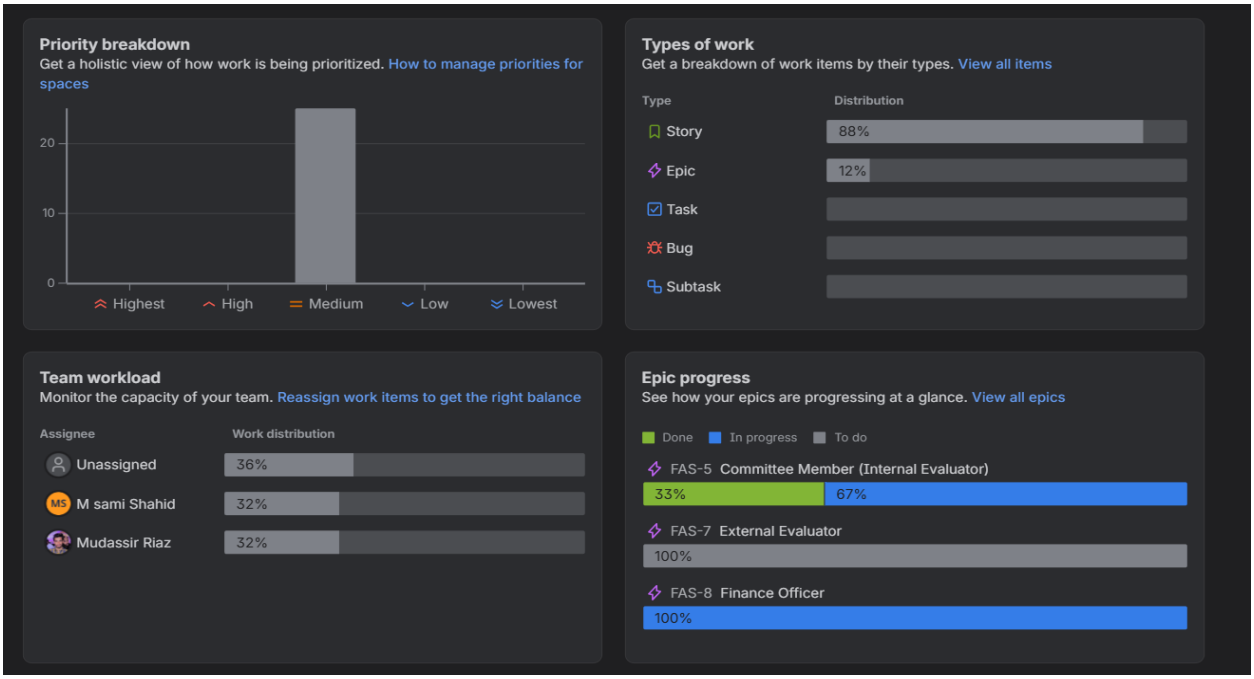


➢ User Stories:

Work	Parent	Description	Assignee	Status	Priority
FAS-9	FAS-1 System Administrator	As a System Admin, I want to register accounts for Students, Teachers, HODs, and External Evaluators (individually or via bulk Excel import), so th...	M sami Shahid	IN PROGRE	Me...
~~FAS-10~~	FAS-1 System Administrator	As a System Admin, I want to view and edit user details (names, emails, Registration IDs) to fix data entry errors.	Mudassir Riaz	DONE	Me...
~~FAS-11~~	FAS-1 System Administrator	As a System Admin, I want to deactivate accounts or reset passwords for users who have left the university or forgotten credentials, ensuring sec...	M sami Shahid	DONE	Me...
FAS-12	FAS-1 System Administrator	As a System Admin, I want to view a chronological log of system activities (logins, deletions, grade changes), so that I can trace unauthorized act...	M sami Shahid	IN PROGRE	Me...
FAS-13	FAS-2 Student	As a Student, I want to complete my profile and search for other students to form a project group (2-3 members), so that we are linked as a single...	M sami Shahid	IN PROGRE	Me...
~~FAS-14~~	FAS-2 Student	As a Student, I want to select a project title and upload my proposal document for Supervisor approval.	M sami Shahid	DONE	Me...
FAS-15	FAS-2 Student	As a Student, if my proposal is deferred, I want to upload a revised version based on feedback without creating a new application.	Mudassir Riaz	IN PROGRE	Me...
~~FAS-16~~	FAS-2 Student	As a Student, I want to upload my SRS, Mid-Term Presentation, and Final Thesis softcopies to the portal before the deadlines.	M sami Shahid	DONE	Me...
FAS-17	FAS-2 Student	As a Student, I want to submit weekly/monthly progress logs digitally, so that my work is documented for grading.	M sami Shahid	IN PROGRE	Me...
FAS-18	FAS-2 Student	As a Student, I want to upload a Bill of Quantities (BoQ) and quotations to request financial assistance for hardware/software.	Mudassir Riaz	IN PROGRE	Me...
FAS-19	FAS-2 Student	As a Student, I want to see a real-time progress bar of my FYP journey (e.g., "Pending Defense," "Grading Complete").	M sami Shahid	IN PROGRE	Me...
FAS-20	FAS-3 Supervisor	As a Supervisor, I want to Accept or Reject student proposal requests, so that I can manage my quota of groups.	Unassigned	IN PROGRE	Me...
~~FAS-21~~	FAS-3 Supervisor	As a Supervisor, I want to review student monthly logs and assign a progress grade (0-10), so that continuous assessment is recorded.	Mudassir Riaz	DONE	Me...
FAS-22	FAS-3 Supervisor	As a Supervisor, I want to digitally sign/approve documents (SRS, Thesis, Funding Requests) so they can move to the next stage (Coordinator/HO...	Unassigned	TO DO	Me...
FAS-22	FAS-3 Supervisor	As a Supervisor, I want to digitally sign/approve documents (SRS, Thesis, Funding Requests) so they can move to the next stage (Coordinator/HO...	Unassigned	TO DO	Me...
~~FAS-23~~	FAS-3 Supervisor	As a Supervisor, I want to verify that a group has made the required corrections after a defense before they are marked as "Complete."	Mudassir Riaz	DONE	Me...
FAS-24	FAS-4 FYP Coordinator	As a Coordinator, I want to set submission deadlines for Proposals, SRS, and Final Thesis, so that the system enforces timely submissions.	Unassigned	TO DO	Me...
FAS-25	FAS-4 FYP Coordinator	As a Coordinator, I want to schedule dates/venues for Proposal, Mid-Term, and Final Defenses and notify all participants automatically.	Unassigned	IN PROGRE	Me...
FAS-26	FAS-4 FYP Coordinator	As a Coordinator, I want to assign specific Internal and External Evaluators to student groups for their defense sessions	Mudassir Riaz	TO DO	Me...
FAS-27	FAS-4 FYP Coordinator	As a Coordinator, I want to manually map a Supervisor to a student group if they are unable to find one themselves.	Unassigned	IN PROGRE	Me...
FAS-28	FAS-4 FYP Coordinator	As a Coordinator, I want to click a button to compile marks from all evaluators and publish the result list (Approved/Deferred/Failed).	Unassigned	IN PROGRE	Me...
FAS-29	FAS-5 Committee Member (Int...	As a Committee Member, I want to view the proposals or thesis documents of assigned groups 48 hours before the defense.	Mudassir Riaz	IN PROGRE	Me...
~~FAS-30~~	FAS-5 Committee Member (Int...	As a Committee Member, I want to score students using a digital rubric (e.g., "Presentation: 4/5") during the defense for instant calculation.	Unassigned	DONE	Me...
FAS-31	FAS-5 Committee Member (Int...	As a Committee Member, I want to enter specific "Required Revisions" into the system so students know exactly what to fix.	Unassigned	IN PROGRE	Me...
FAS-32	FAS-6 Head of Department (H...	As an HOD, I want to receive alerts if a group misses two consecutive monthly reports so I can issue a warning.	Unassigned	TO DO	Me...
FAS-33	FAS-6 Head of Department (H...	As an HOD, I want to review and approve the budget amount for student projects so the Finance Department can process it.	Mudassir Riaz	IN PROGRE	Me...
FAS-34	FAS-6 Head of Department (H...	As an HOD, I want to review the final compiled marks for fairness/variance before they are officially published.	Unassigned	TO DO	Me...
FAS-35	FAS-7 External Evaluator	As an External Evaluator, I want to access student project files via a secure login to prepare for the Final Viva.	M sami Shahid	TO DO	Me...
FAS-36	FAS-7 External Evaluator	As an External Evaluator, I want to input my marks for the Final Defense directly into the system to be averaged with internal scores.	Mudassir Riaz	TO DO	Me...
FAS-37	FAS-8 Finance Officer		M sami Shahid	IN PROGRE	Me...

➢ Summary:





Google Drive Link Containg User Stories in Exel:

https://drive.google.com/drive/folders/1DobHgOf1wtjls2RFhDXtuy4YscRvPc8T?usp=sharing

6. *Non Functional Requirements*

6.1. Performance Requirements

- The system shall support 500 concurrent users during submission deadlines.
- Page load time should be under 2 seconds for standard forms.

6.2. Safety Requirements

- The system must perform regular database backups (daily).
- Deleted data should be soft-deleted (archived) rather than permanently removed immediately.

6.3. Security Requirements

- **Authentication:** All users must authenticate via secure login.
- **Authorization:** RBAC must prevent Students from accessing Supervisor functions.
- **Data Encryption:** Passwords must be hashed using BCrypt.

6.4. **Software Quality Attributes**

- **Reliability:** The system should have 99.9% uptime during the semester.
- **Maintainability:** The backend code should follow MVC architecture for easy updates.

6.5. **Business Rules**

- **BR-01:** A student cannot belong to more than one group.
- **BR-02:** A Supervisor cannot exceed their assigned quota (e.g., 5 groups).
- **BR-03:** A project cannot be archived until all Forms(A-D) are signed

SYSTEM DESIGN SPECIFICATION

1. Introduction

The system architecture defines the high-level structure of the FYP Management System, identifying the software components, their external properties, and their relationships. This section provides a blueprint for the system's organization, ensuring that the design meets the performance, security, and scalability requirements defined in the SRS. It serves as the foundation for all subsequent development activities.

1.1. Architectural Design

The FYP Management System utilizes a **Three-Tier Client-Server Architecture** (also known as Multi-Tier Architecture). This pattern separates the system into three logical and physical computing tiers:

1. **Presentation Tier (Client):**
    - **Technology:** React.js (Single Page Application).
    - **Role:** Handles user interaction, UI rendering, and client-side state management. It communicates with the server strictly via HTTP APIs.
2. **Application Tier (Business Logic):**
    - **Technology:** ASP.NET Core Web API.

- **Role:** Processes business rules (e.g., grading logic, workflow transitions), handles authentication/authorization, and acts as the bridge between the client and the database.

3. **Data Tier (Storage):**
   - **Technology:** Microsoft SQL Server.
   - **Role:** Stores and manages all persistent data, ensuring data integrity through relational constraints.

1.2. Justification for Architectural Design

The Three-Tier architecture was selected for the following reasons:

- **Separation of Concerns:** The User Interface is completely decoupled from the Data Storage and Business Logic. This allows the frontend (React) and backend (.NET) teams to work independently without blocking each other.
- **Scalability:** Each tier can be scaled independently. If the frontend traffic increases, the static assets can be served via a CDN. If processing load increases, the API server can be horizontally scaled.
- **Security:** The database is not exposed directly to the client. The Application Tier acts as a gatekeeper, validating all inputs and enforcing Role-Based Access Control (RBAC) before any data is queried or modified.
- **Maintainability:** Business logic is centralized in the API. If a grading rule changes, it only needs to be updated in the Application Tier, without requiring a redeployment of the frontend or database schema changes.

1.3. Design Patterns

The system implements several industry-standard design patterns to ensure code quality and reusability:

Backend Patterns (.NET Core)

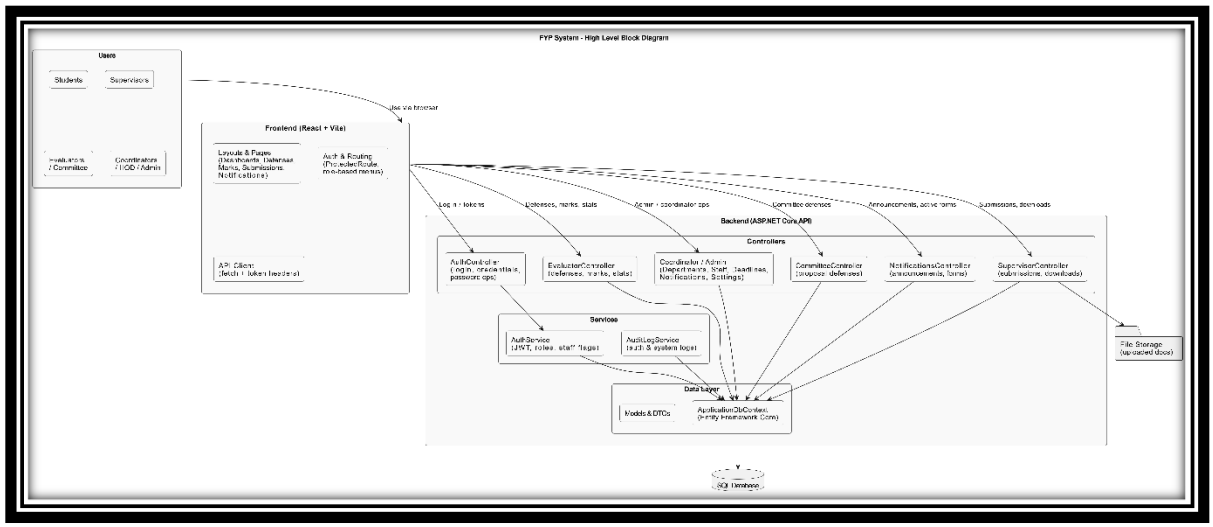
- **Repository Pattern:** Used to abstract data access logic. The controllers do not access the DbContext directly; instead, they communicate with Repositories (e.g., IGroupRepository), making the code unit-testable.

- **Dependency Injection (DI):** ASP.NET Core's built-in DI container is used to inject services (e.g., IEmailService, ITokenService) into controllers, promoting loose coupling.
- **Data Transfer Object (DTO):** DTOs are used to pass data between the API and the client, preventing the accidental exposure of sensitive internal database entities (like password hashes).
- **Singleton Pattern:** Used for services that must maintain a single instance throughout the application lifecycle, such as the ConfigurationService.

Frontend Patterns (React)

- **Higher-Order Components (HOC):** Used for protected routes (e.g., ProtectedRoute) to check if a user is authenticated and has the correct role before rendering a page.
- **Observer Pattern:** Implemented via React Context API or state management libraries to update UI components automatically when the application state changes (e.g., updating the notification bell when a new alert arrives).

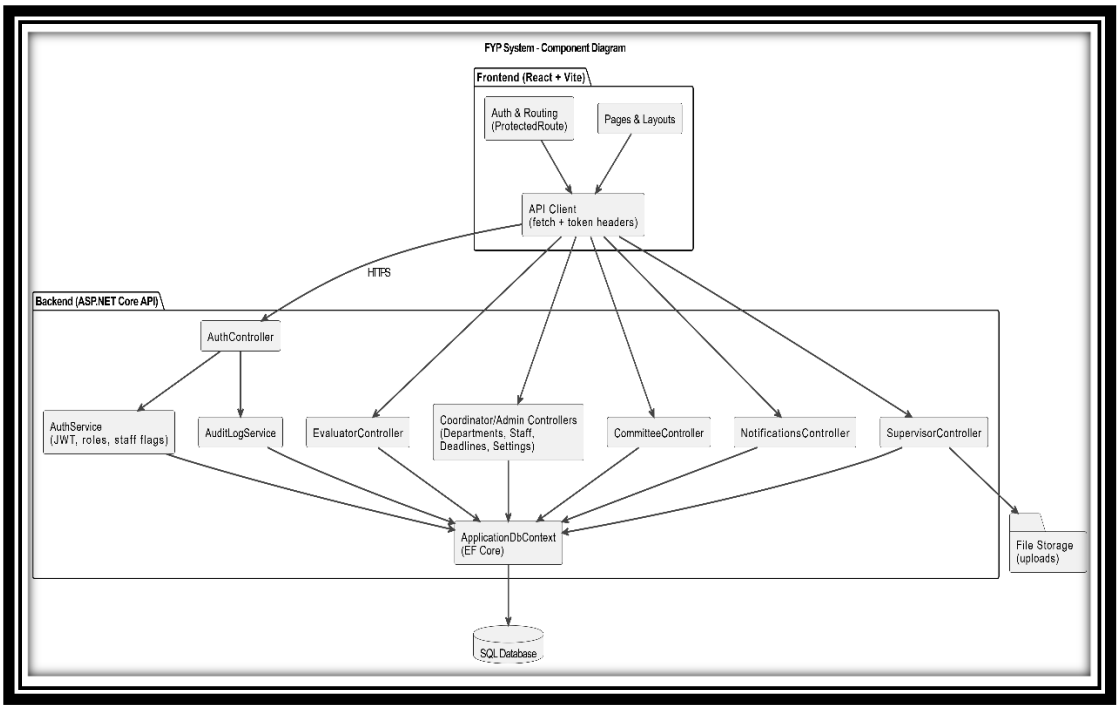
1.4. Block Diagram



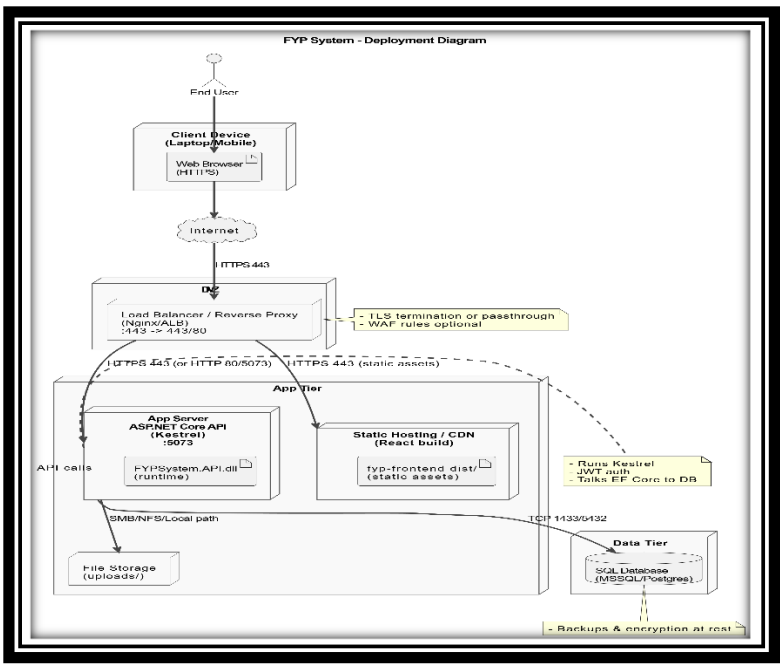
2. Detailed Design

The detailed design bridges the gap between the high-level architecture and the implementation. It defines the specific logical structure of the code, the database schema, and the behavior of the system components. This section serves as the primary reference for developers during the coding phase.

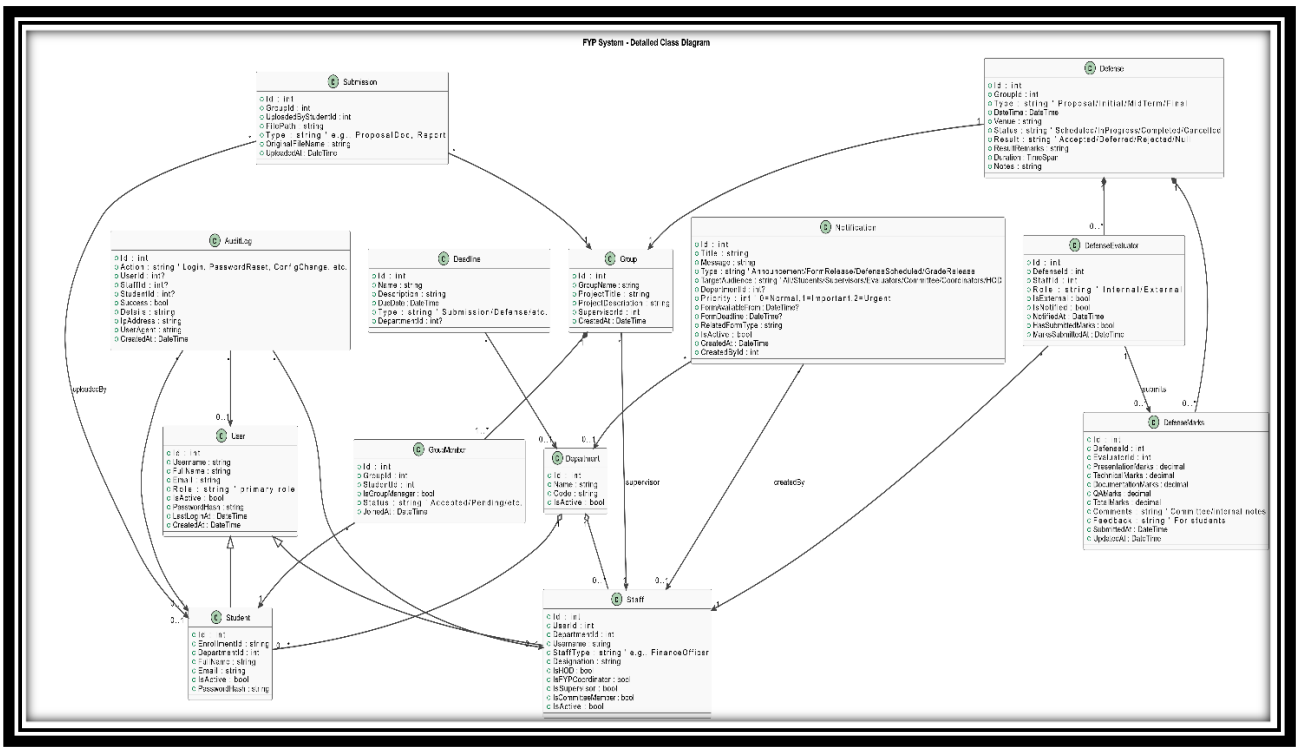
2.1. Component Diagram



2.2. Deployment Diagram

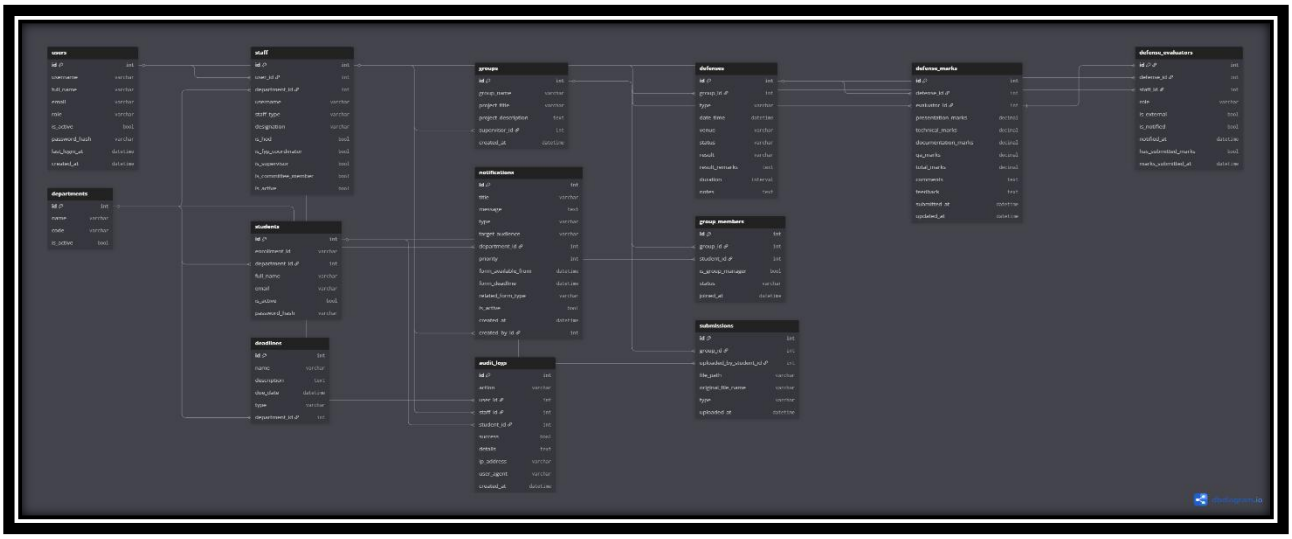


2.3. Class Diagram



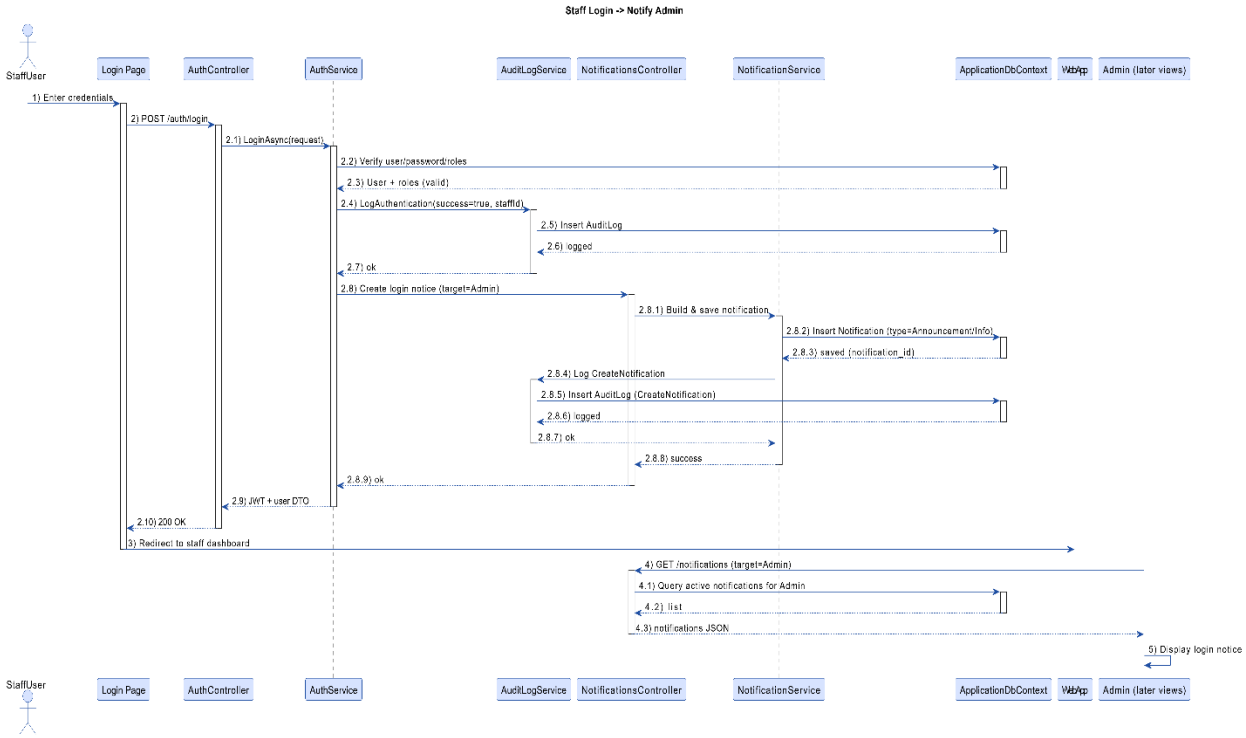
2.4. ERD Diagram

https://dbdiagram.io/d/693980e3e877c630744d3e5f

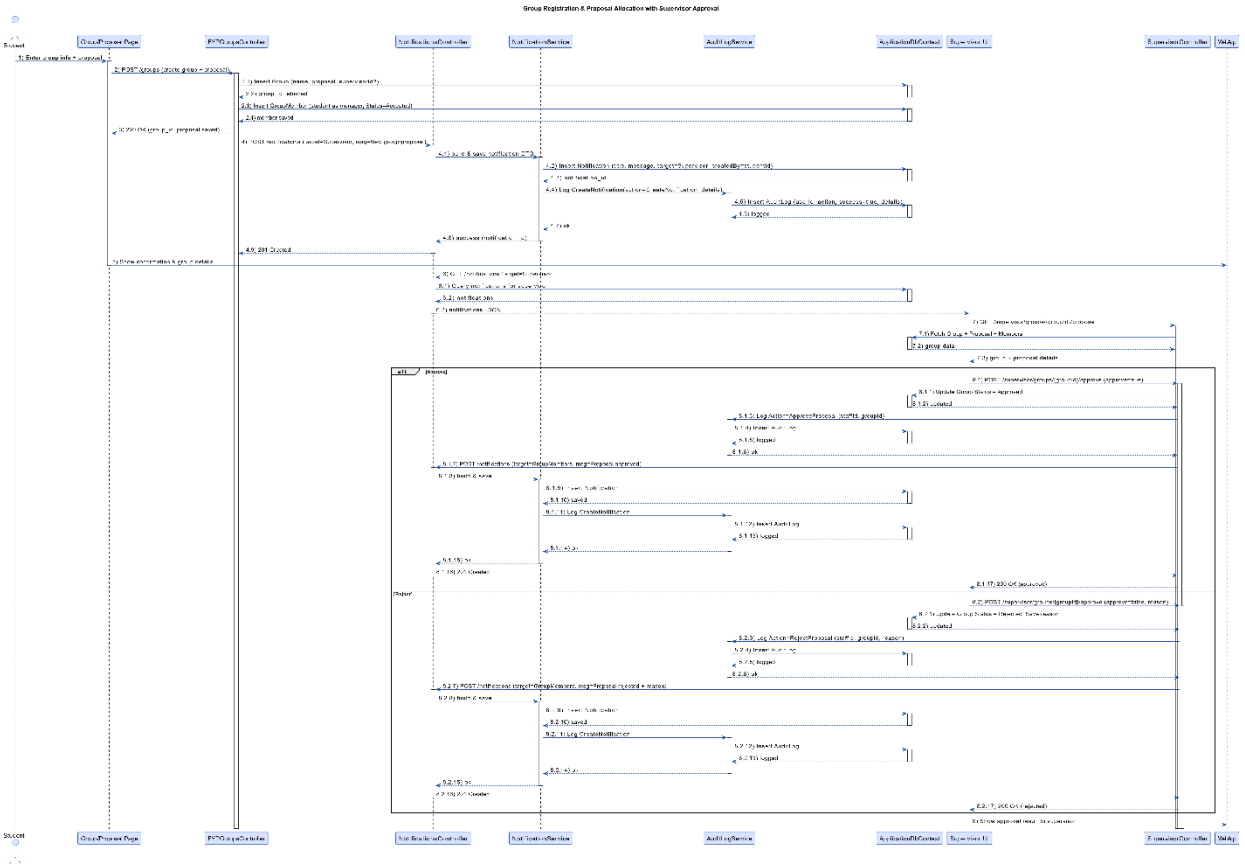


2.5. Sequence Diagrams

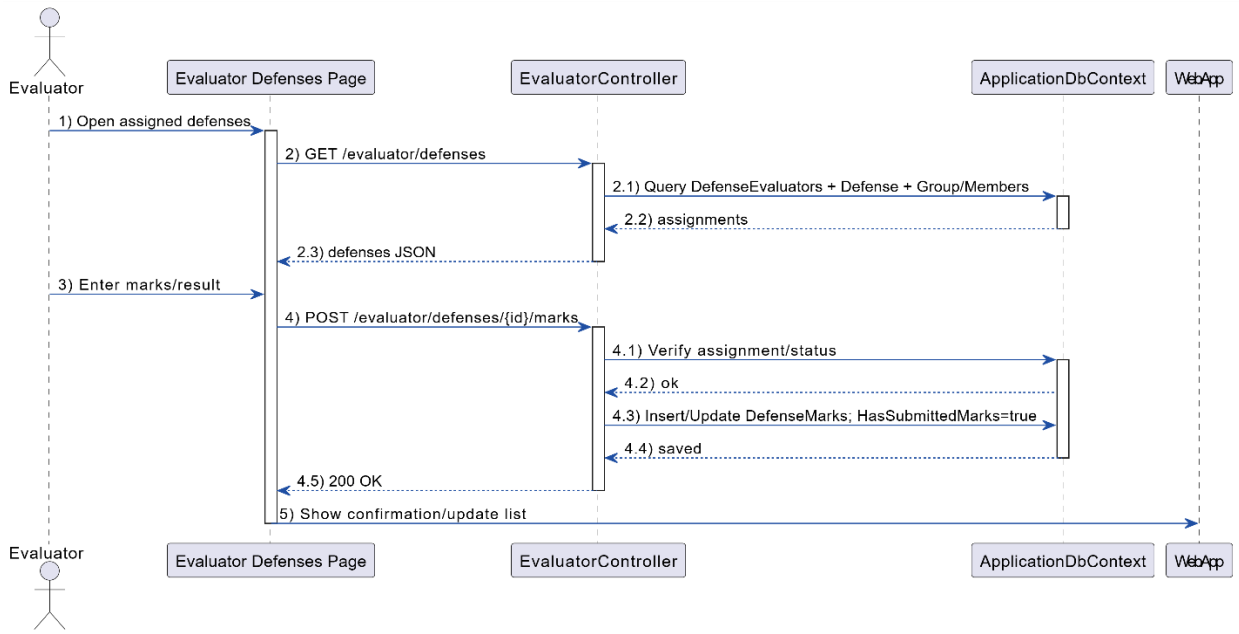
Logs In:



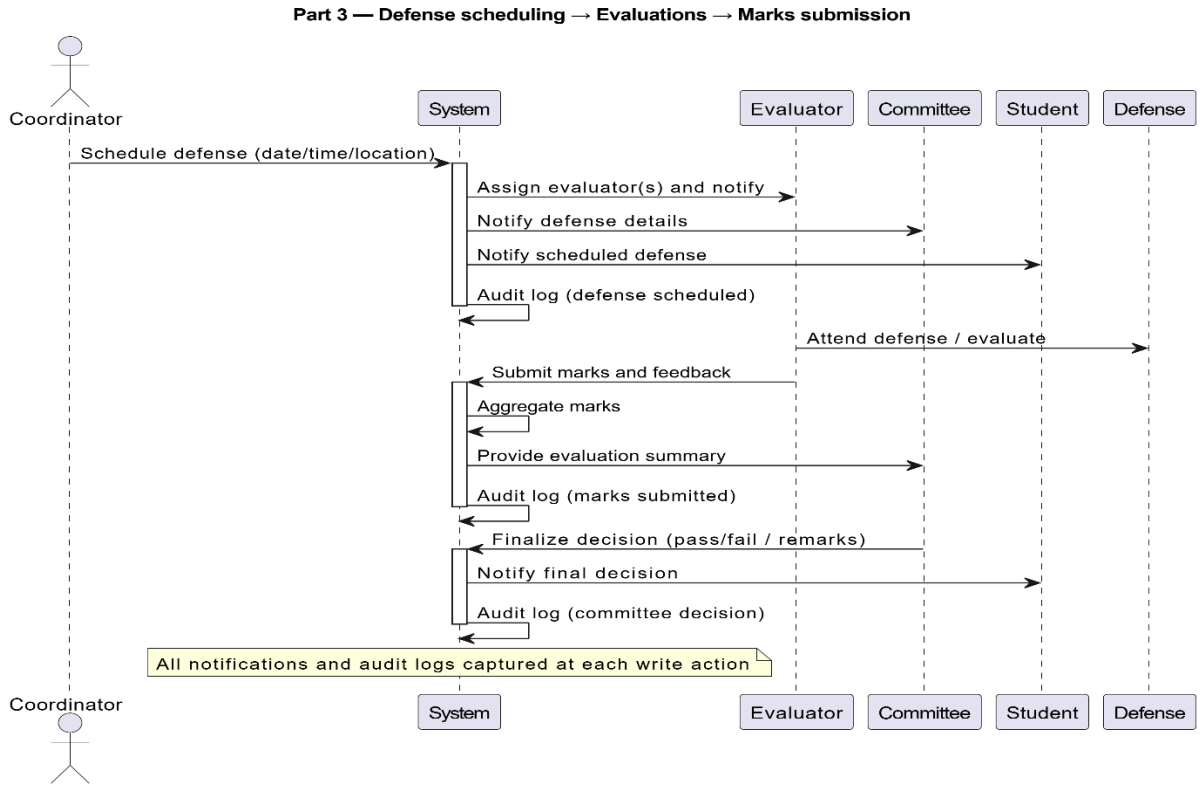
Group Registration & Proposal Allocation:



Evaluator Views Defense and Marks:

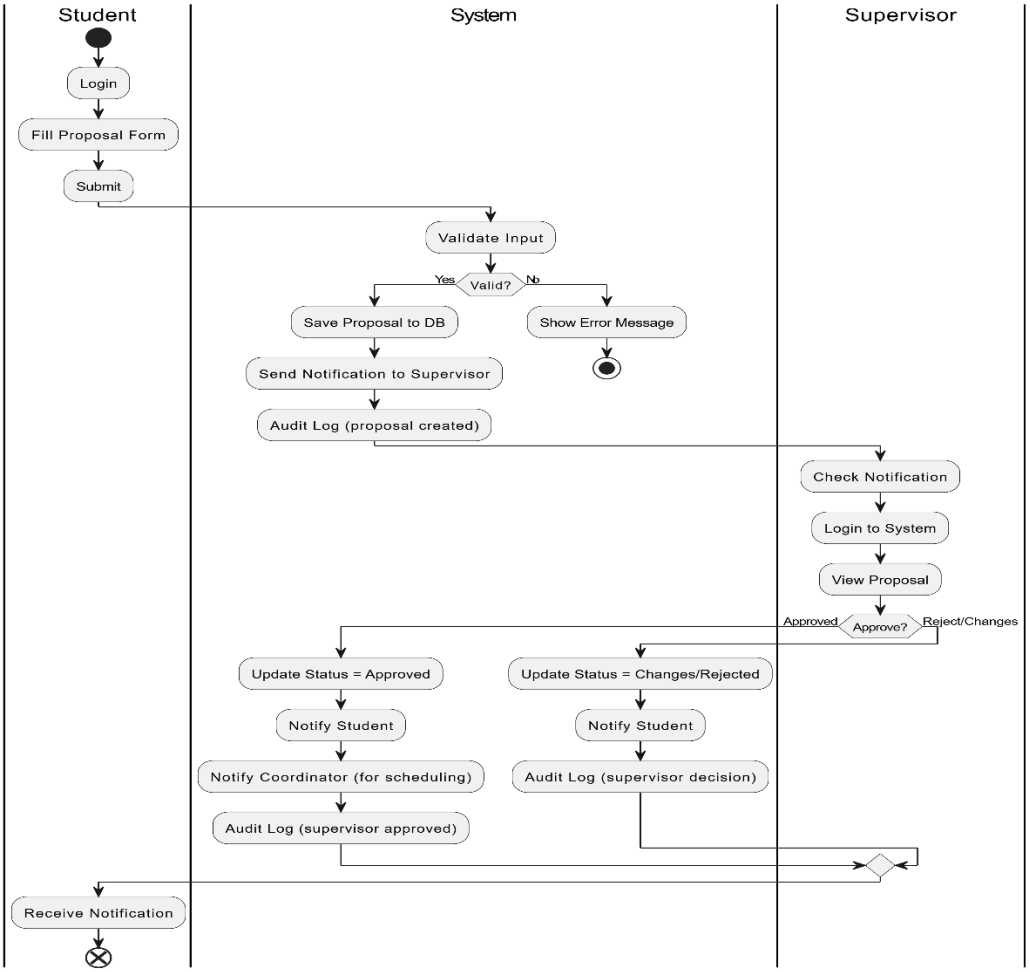


Defense Scheduling:

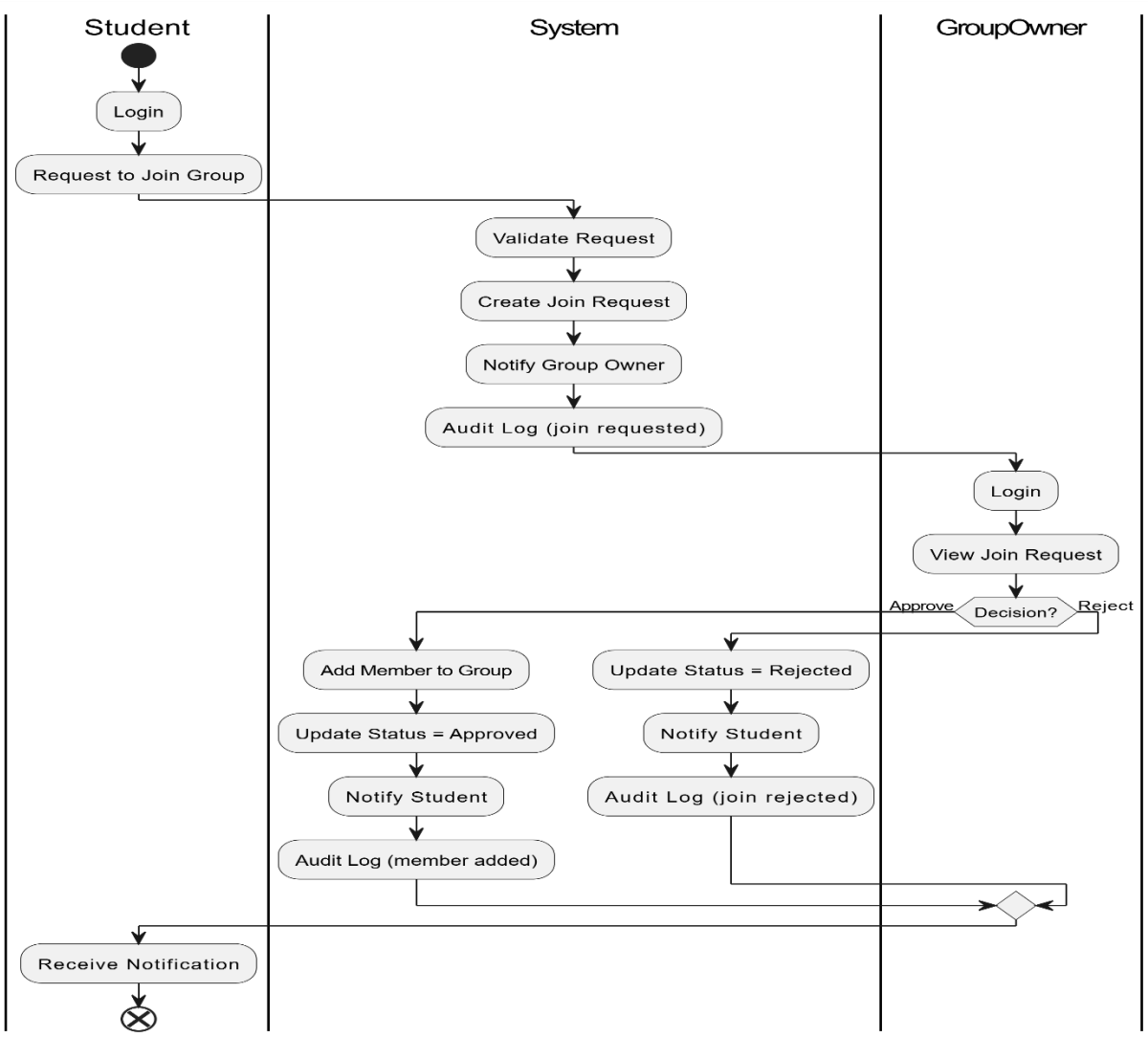


2.6. Activity Diagrams

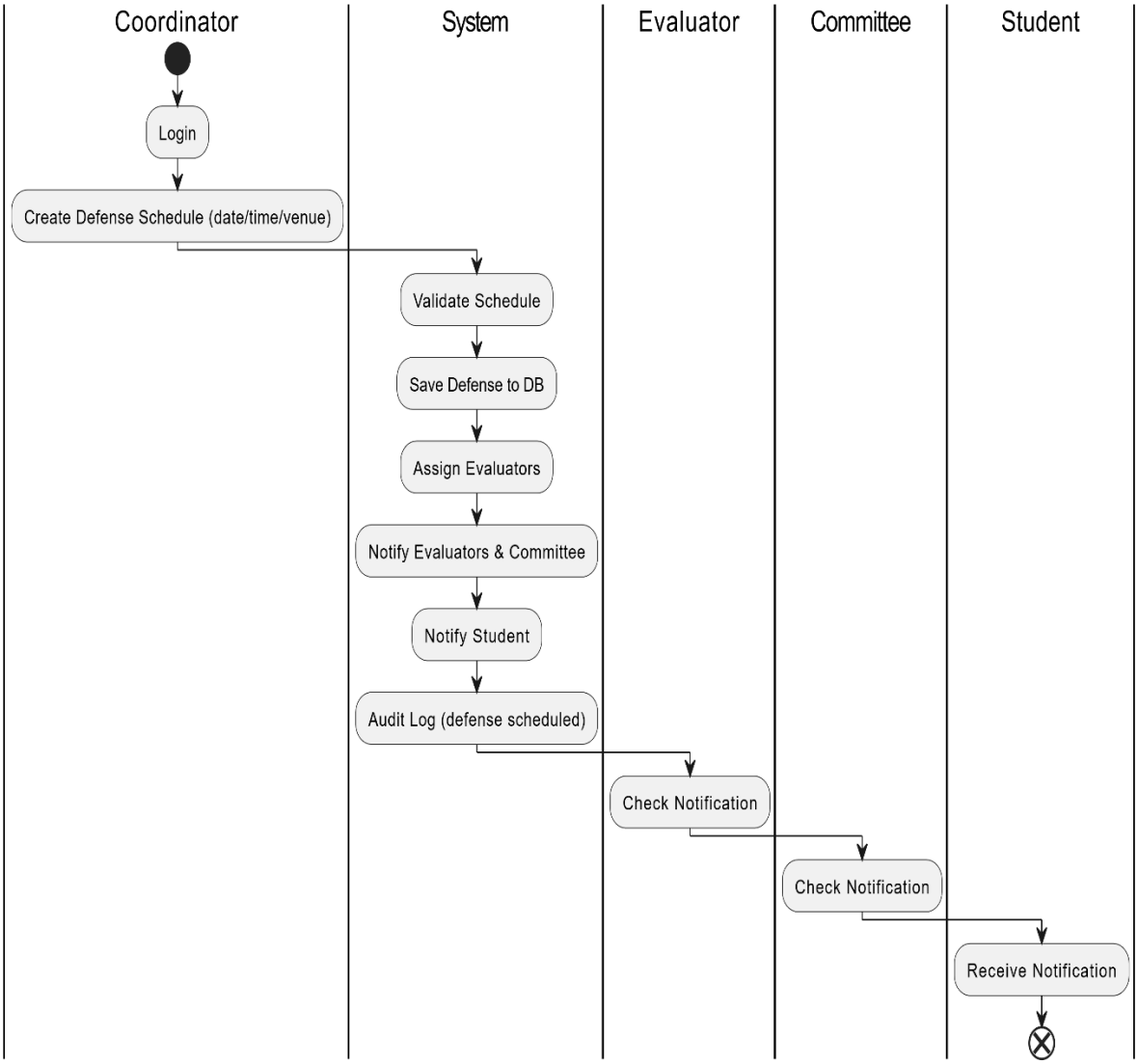
Proposal:



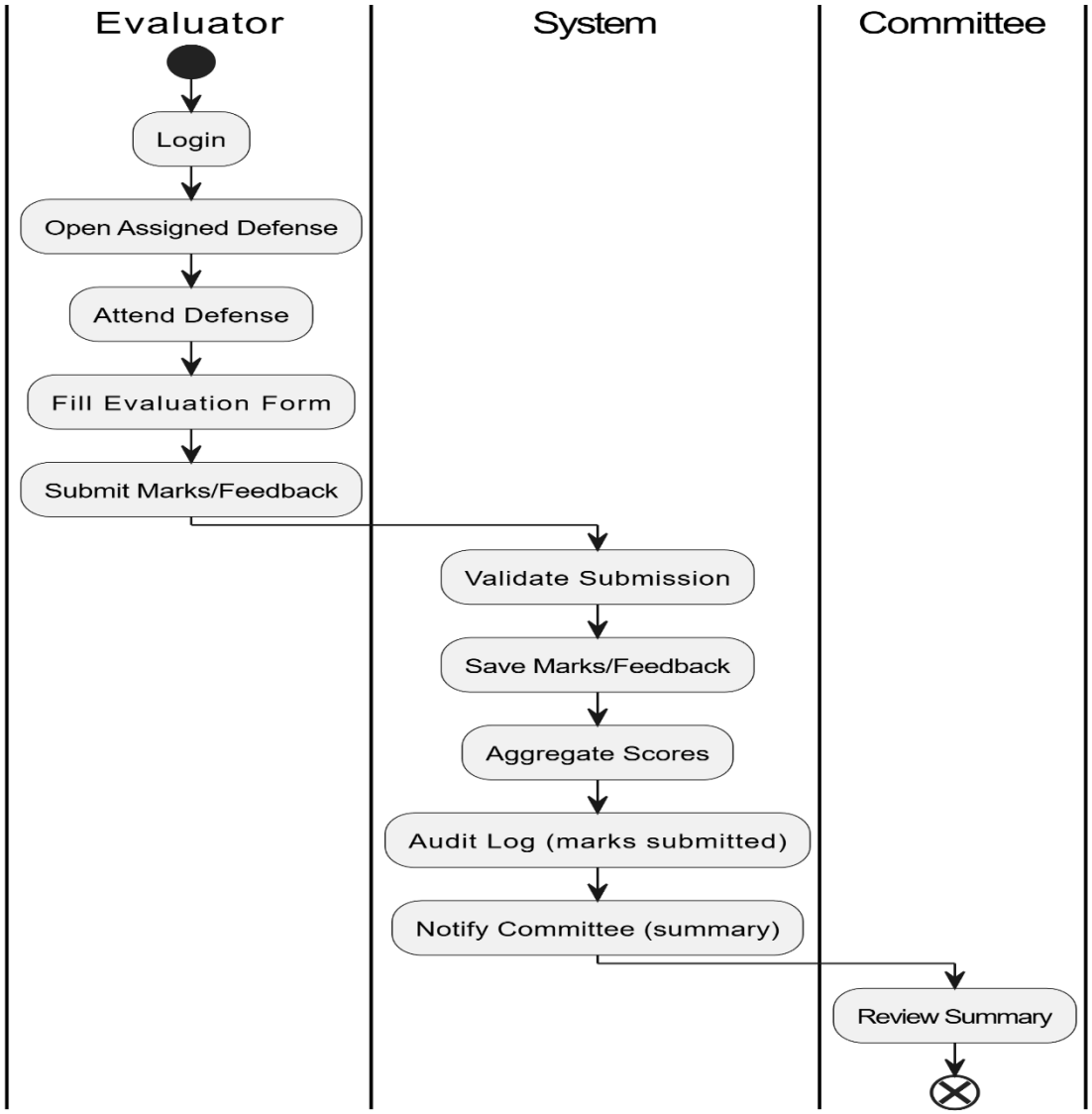
MemberShip:


Student
System
GroupOwner
Login
Request to Join Group
Validate Request
Create Join Request
Notify Group Owner
Audit Log (join requested)
Login
View Join Request
Approve
Decision?
Reject
Add Member to Group
Update Status = Approved
Notify Student
Audit Log (member added)
Update Status = Rejected
Notify Student
Audit Log (join rejected)
Receive Notification

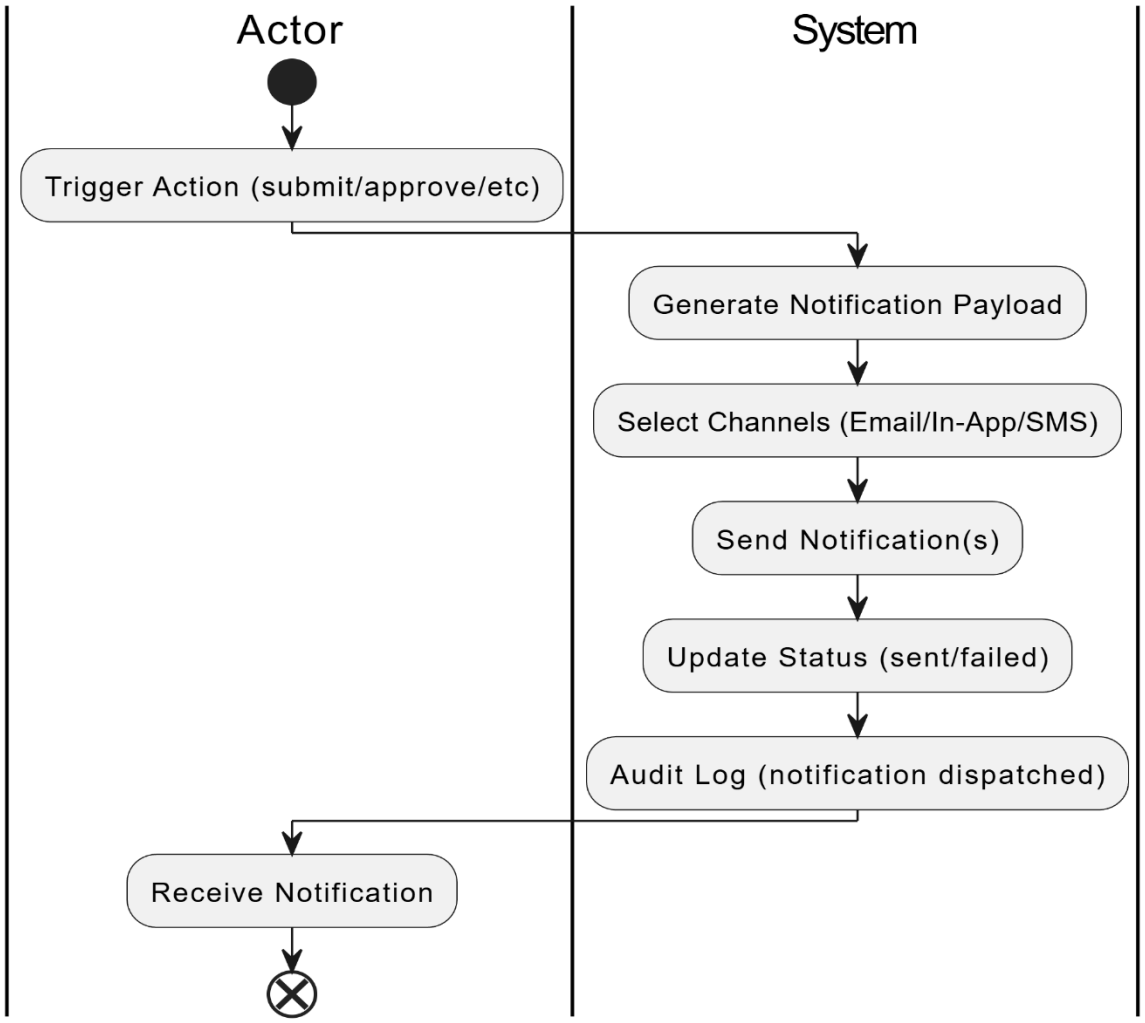
Defense Scheduling:



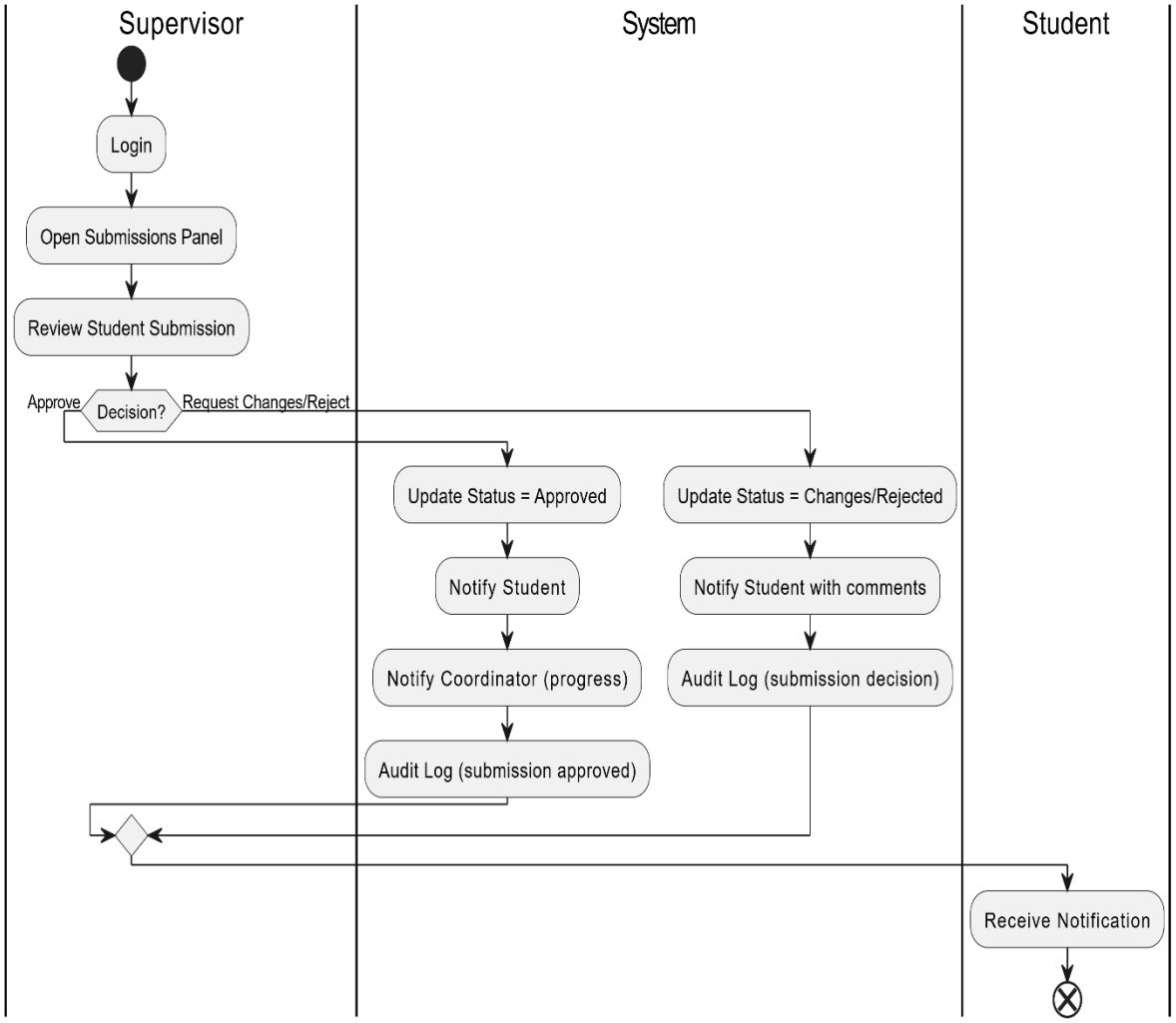
Evaluating Marks:



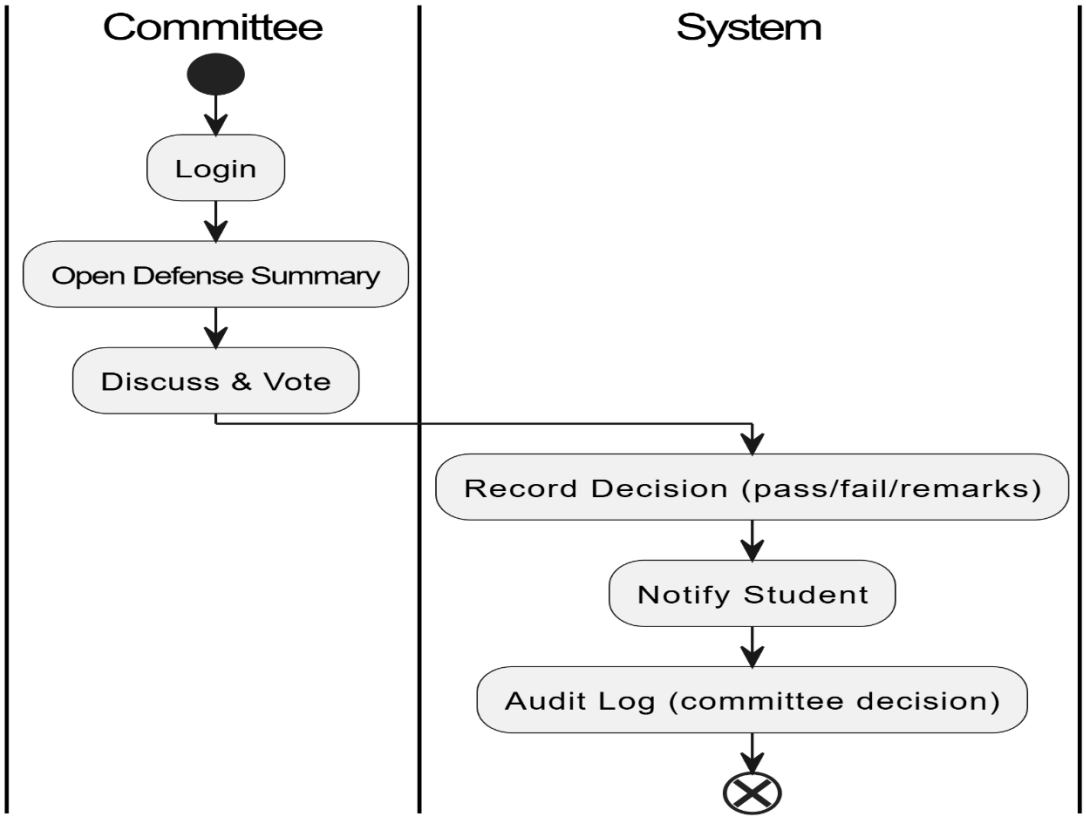
Notifications Activity:



Supervisor Submission:



Committee Activity:



2.7. Design Constraints

This section outlines key technical constraints for the FYP Management System, accompanied by code snippets illustrating the required implementation patterns.

1. **Security & Authentication**

- **Authentication Mechanism:** System must enforce **JWT-based authentication**. Tokens must include encrypted role claims to facilitate stateless authorization checks.
- **Authorization:** All state-changing API endpoints (POST, PUT, DELETE, PATCH) must utilize [Authorize(Roles = ...)] attributes.
- **Credential Storage:** Passwords must never be stored in plain text; use **BCrypt** or **PBKDF2** with per-user salting.
- **Transport Security: HTTPS** is mandatory for all traffic. HTTP requests must be redirected to HTTPS.
- **CORS Policy:** Cross-Origin Resource Sharing must be restricted strictly to known client origins (e.g., the specific React frontend URL) and not set to *.
- **Rate Limiting:** Authentication and public endpoints must implement rate limiting (e.g., max 5 login attempts per minute) to prevent brute-force attacks.

2. **Data Integrity & Storage**

- **Transactional Integrity:** Multi-step write operations (e.g., Scheduling a defense + Sending notifications + Locking the slot) must be wrapped in a Database Transaction.
- **Schema Constraints:** Database must enforce Foreign Keys and Unique Constraints on critical entities (Users, Groups, Defense Slots) to prevent orphaned or duplicate data.
- **Auditability:** Records must utilize **Soft Delete** (IsDeleted flag) rather than physical deletion to maintain historical integrity.
- **Validation:** Input validation must occur on both the Client (for UX) and Server (for security). API DTOs must use Data Annotations or FluentValidation.

3. **Availability & Performance**

- **Latency Targets:** API endpoints must respond within **500ms (p95)** under normal load.

- **Pagination:** All list/collection endpoints must support server-side pagination (limit/offset or cursor-based) to prevent large payload transfer.
- **Indexing Strategy:** Database indexes must be applied to high-frequency lookup fields: UserId, GroupId, DefenseId, and Role.
- **Asynchronous Processing:** Long-running tasks, specifically email/notification dispatch, must be offloaded to a background queue (e.g., BackgroundService or Hangfire) to ensure non-blocking API responses.

4. Audit & Compliance

- **Logging Scope:** The system must log all critical business actions: Authentication events, Approvals, Defense Scheduling, Document Submissions, Mark Entries, and Notification dispatches.
- **Log Format:** Audit logs must be time-synced to **UTC** and include the Actor ID, IP Address, Action Type, and Timestamp.
- **Immutability:** Audit records must be read-only and immutable once written.

5. Notifications & Communication

- **Delivery Mechanism:** Notifications must be processed via a non-blocking queue.
- **Reliability:** The notification service must implement a **Retry Pattern** with exponential backoff for failed deliveries (e.g., SMTP timeout).
- **Failure Handling:** Messages failing after maximum retries must be moved to a Dead-Letter Queue (DLQ) for administrator review.
- **State Tracking:** In-app notifications must persist read/unread state per user.

6. Scheduling & Time Management

- **Time Storage:** All timestamps in the database must be stored in **UTC**.
- **Time Display:** Timestamps must be converted to the user's local timezone by the Frontend before display.
- **Conflict Detection:** The system must enforce server-side validation to prevent overlapping defense slots for the same Venue or Evaluator.
- **Cutoff Enforcement:** Submission deadlines must be enforced server-side based on the server time, not client time.

7. **File Handling**

- **Validation:** File uploads must be validated for MIME type (signature check) and file size limits.
- **Security Scanning:** Uploads should be passed through a virus scanner if infrastructure permits.
- **Storage Location:** Files must be stored outside the webroot (wwwroot) or in cloud storage (e.g., Azure Blob).
- **Access Control:** Files must not be publicly accessible via static URLs; access requires an authorized API call to generate a temporary download stream or signed URL.

8. **Architecture & Extensibility**

- **Separation of Concerns:** Business logic must reside in the **Service Layer**, not Controllers. Controllers should only handle HTTP concerns (request parsing, response formatting).
- **Configuration:** Feature flags and infrastructure settings must be externalized in appsettings.json or Environment Variables.
- **Hardcoding:** "Magic strings" for Roles or Statuses must be avoided; use centralized **Enums** or Constant classes.
- **DTO Isolation:** Domain entities must be mapped to specific DTOs (Data Transfer Objects) for API responses; Entities should not be exposed directly.

9. **Frontend Build & UX**

- **Code Quality:** Build pipeline must enforce Linting and Formatting rules; builds should fail on blocking console errors.
- **Configuration:** API Base URLs must be injected via Environment Variables to support distinct Dev, Stage, and Prod environments.
- **Session Management:** The frontend must handle 401 Unauthorized responses gracefully by clearing local state and redirecting to the login page.
- **Data Handling:**
    - Defensive handling of CamelCase/PascalCase JSON payloads.
    - Strict null/undefined guards in rendering logic.
    - Use stable keys (IDs) for list rendering.

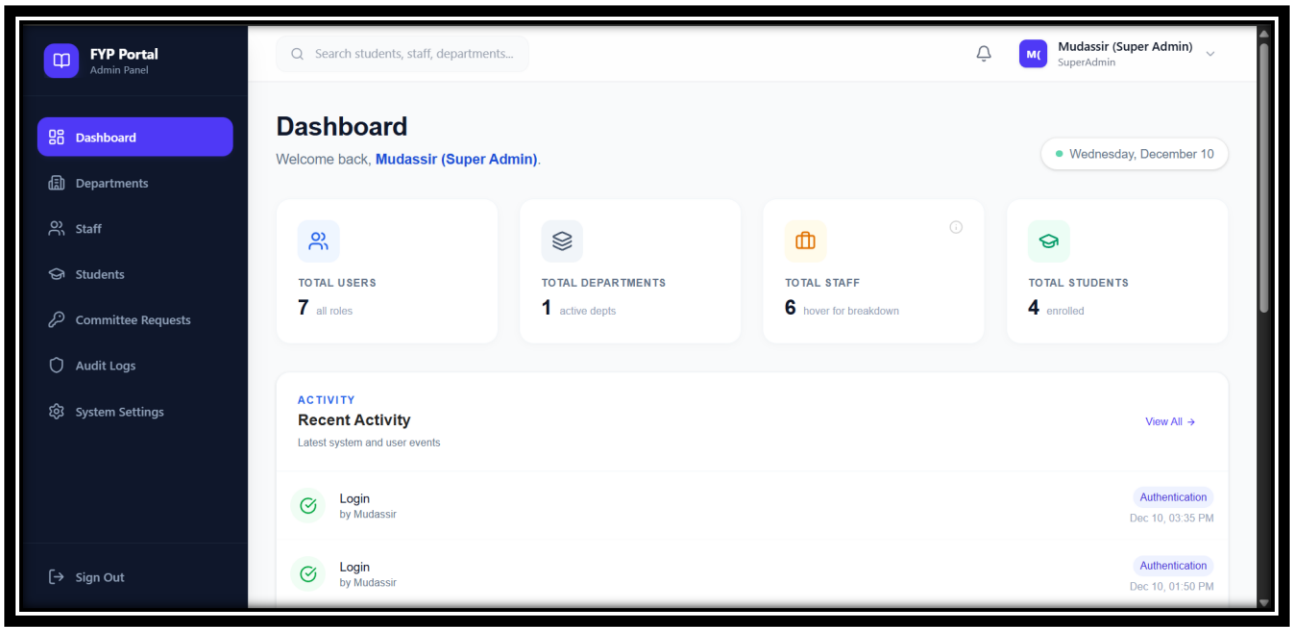
- Optimistic UI updates should be restricted to reversible actions (e.g., "Like", "Mark Read").

10. Testing Strategy

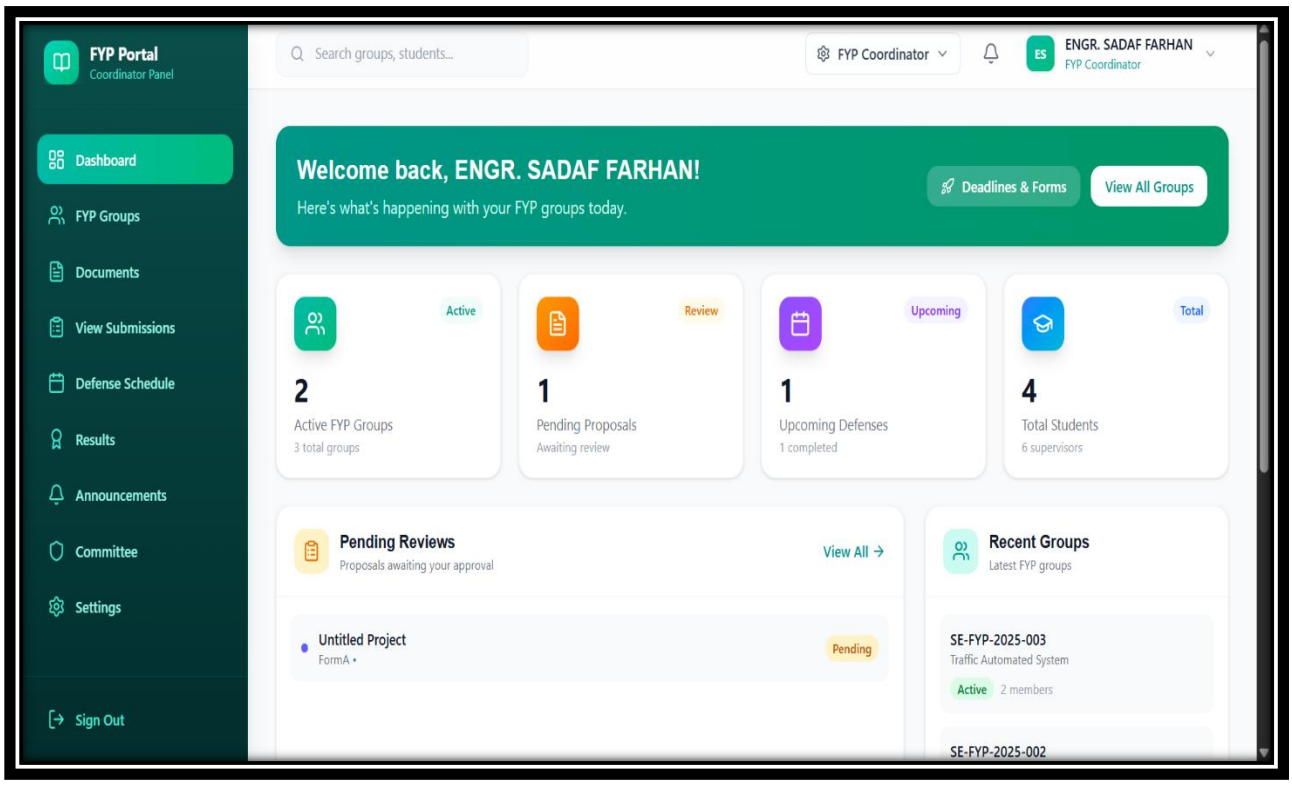
- **Unit Tests:** Required for all Services, Utility classes, and Mappers.
- **Integration Tests:** Required for critical business flows: Authentication, Proposal Submission/Approval, Defense Scheduling, and Marks Submission.
- **Contract Tests:** Notification payloads and API contracts must be verified to ensure frontend/backend alignment.

3. Detailed UI Designs

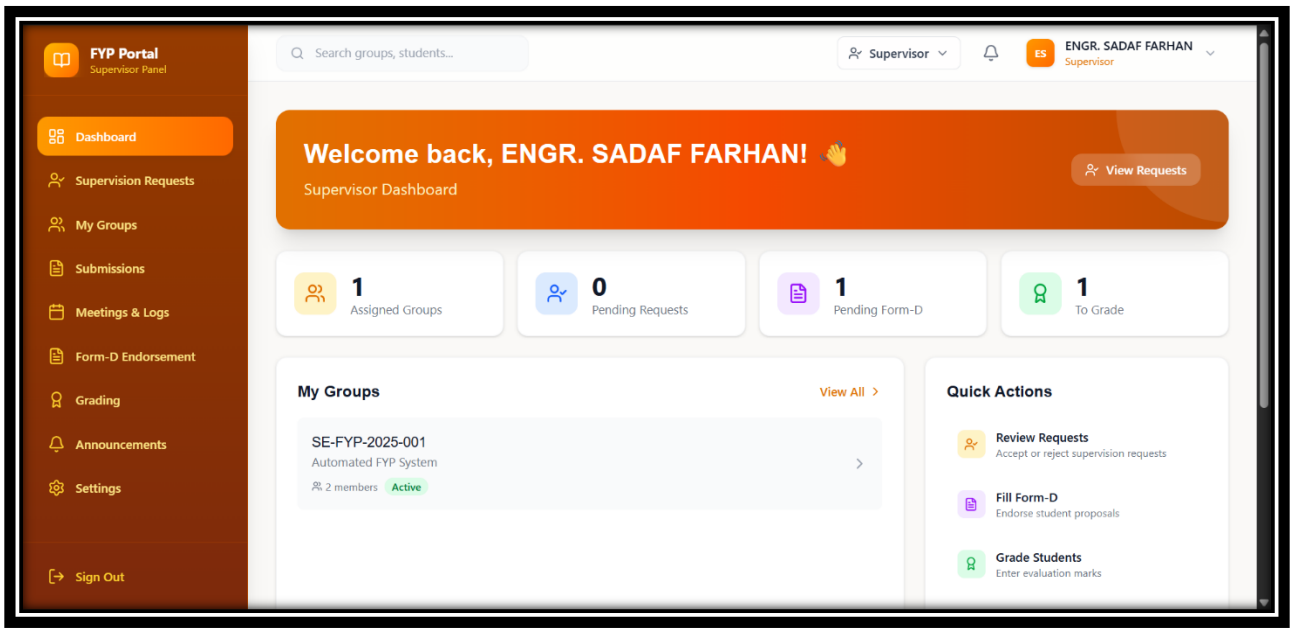
Admin Dashboard:



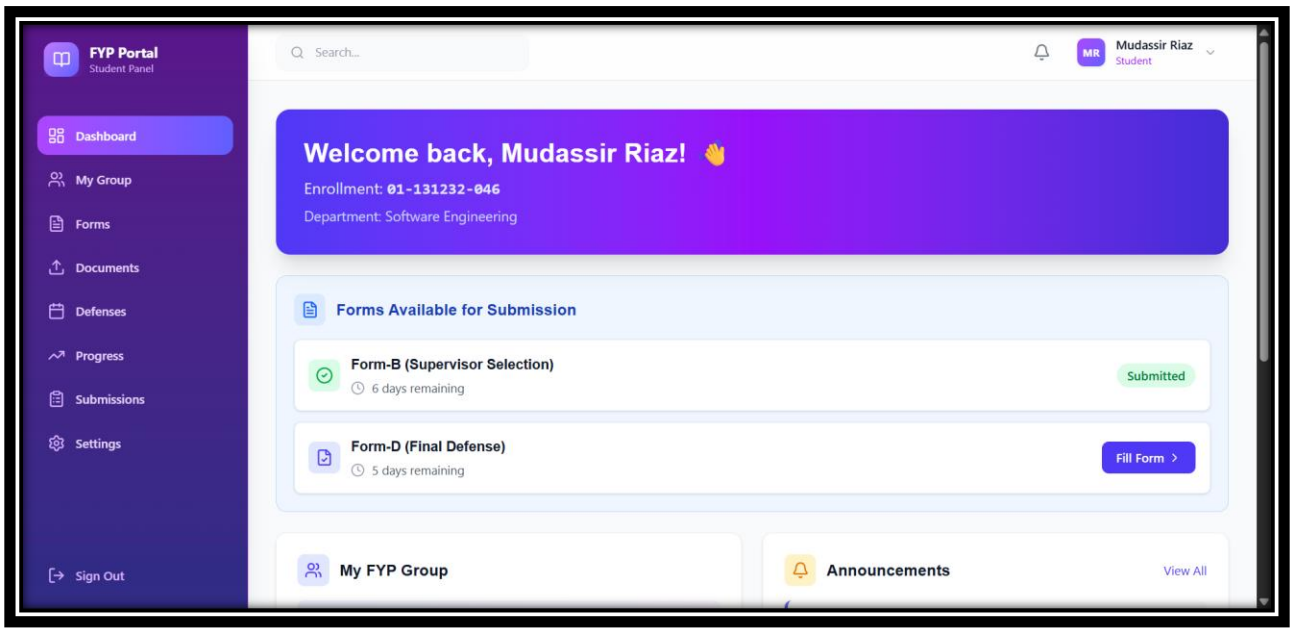
Coordinator Dashboard:



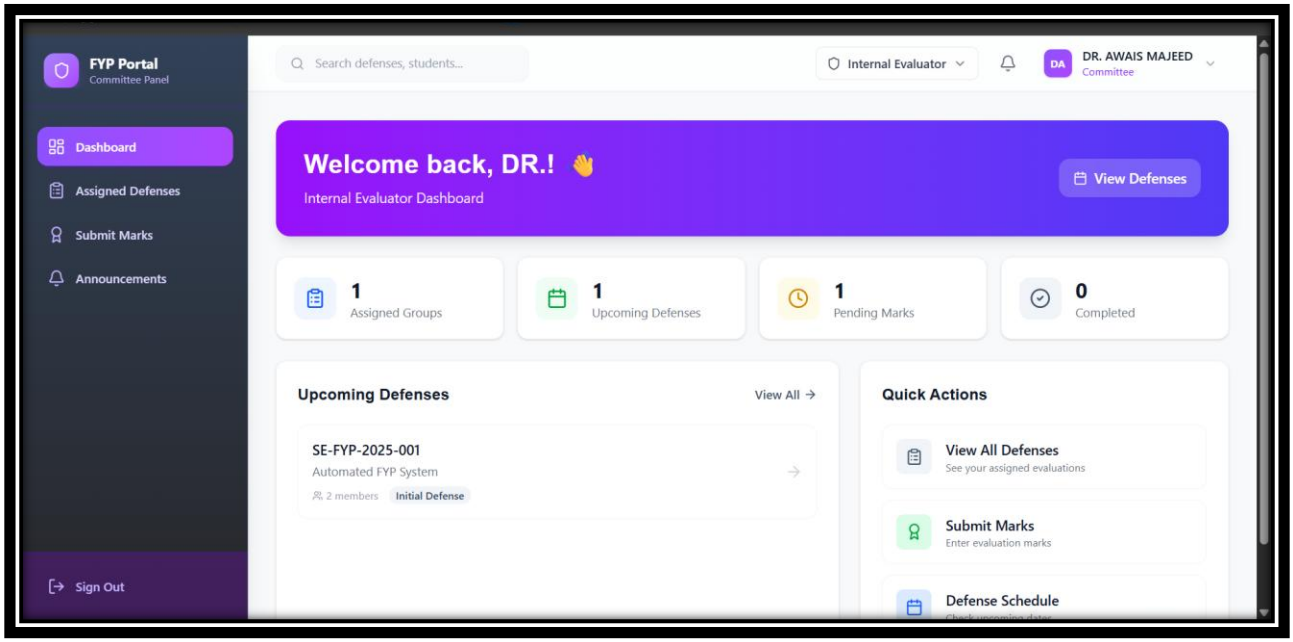
Supervisor Dashboard:



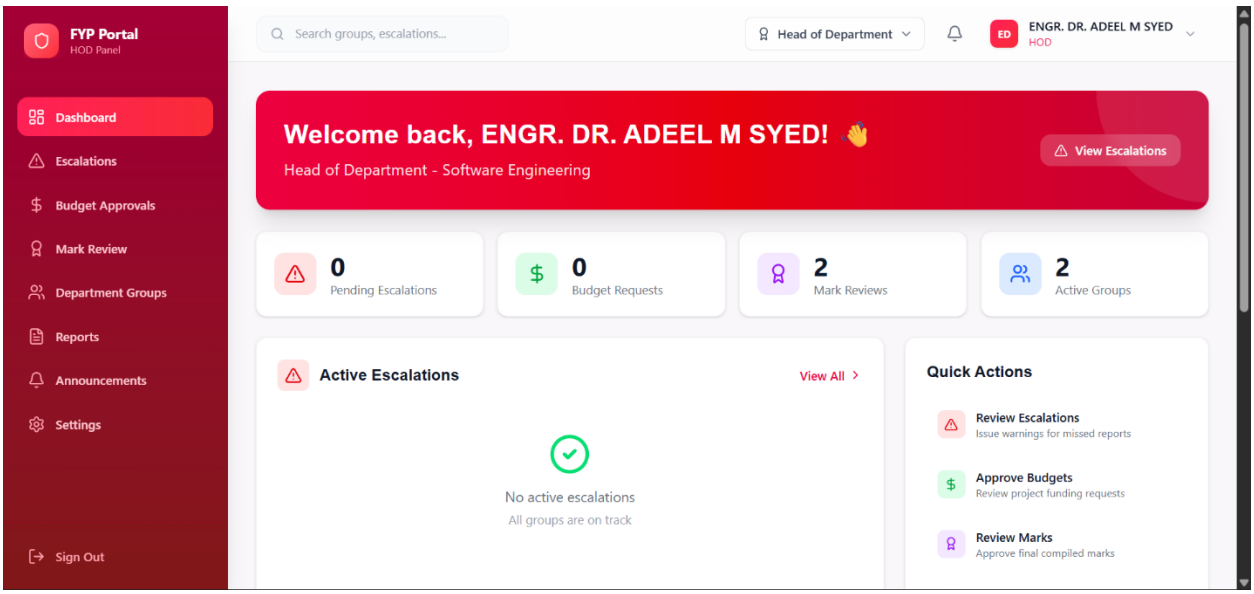
Student Dashboard:



Evaluator Dashboard:



HOD Dashboard:



Github Link

https://github.com/mudassirriaz64/FYP-SYSTEM